

AMENDMENTS TO THE SPECIFICATION

In the Specification:

Please replace the paragraph beginning on page 9, line 2, which starts with “Fig. 3 illustrates”, with the following amended paragraph:

Fig. 3 illustrates an example of a string template page 70. The string template page 70 includes four strings and three variable arguments within the strings. A first[[s]] string 72 includes a meta-tag labeled “BREAK” with a unique identifier name labeled “GUID1”. The first string 72 is the header string. A second string 74 includes another meta-tag labeled “BREAK” with a unique identifier name labeled “GUID2”. The second string 74 also includes two meta-tags labeled “INSERT” with corresponding argument numbers “ARG1” and “ARG2” within the HTML string. A third string 76 includes a meta-tag labeled “BREAK” with a unique identifier name labeled “GUID3”. A fourth string 78 includes another meta-tag labeled “BREAK” with a unique identifier name labeled “GUID4”. The fourth string 78 also includes a meta-tag labeled “INSERT” with an argument numbers “ARG3” within the HTML text of the string 78. The “INSERT” meta-tags define the beginning and end location of each string and the “INSERT” meta-tags define the location for inserting the different arguments by number. The executable component 54 can employ the “BREAK” and “INSERT” meta-tags to load the text constant information of the page 70 into memory 56 for use at runtime.

Please replace the paragraph beginning on page 9, line 26, which starts with “Figs. 4 and 5 illustrate”, with the following amended paragraph:

Figs. 4 and 5 illustrate an example of employing intelligent IDs in code of the executable component referencing the string template page 70 illustrated in Fig. 3. A data structure portion 80 includes a page data structure for the string template page 70 with a corresponding page ID. The page data structure includes a page name or ID and a pointer to the location of the page in memory. A plurality of string data structures are provided for the string template page 70, which includes a first named “HEADER” and three remaining strings named “GUID2”, “GUID3” and “GUID4”. Each string data structure includes a pointer to the corresponding string in memory.

Additionally, a list of argument number data structures ~~is~~ ~~are~~ provided for the string template page 70. The argument numbers are labeled ARG1, ARG2 and ARG3. Each argument data structure includes a pointer to the corresponding argument in memory. In the example of Fig. 4, none of the IDs of the page, strings and arguments have been referenced and all of the pointers are set to "NULL". Fig. 5 illustrates the data structure portion 80 once the IDs have been referenced. After all of the IDs of the page, strings and arguments have been referenced, then all of the pointers have a pointer value for the data structure portion 80. In all future references to the IDs of the page strings and arguments the corresponding pointer can be used directly. It is to be appreciated that although in the present example, a single list of arguments for the string template page 70 is illustrated for simplicity purposes, a list of arguments may be provided for each string in the string template page 70.

Please replace the paragraph beginning on page 11, line 5, which starts with "Fig. 7 illustrates", with the following amended paragraph:

Fig. 7 illustrates a methodology for the operation of the compiled binary executable code. At step 160, the compiled binary executable code loads the string template pages up into temporary memory or the like. The executable code then parses the string template pages and determines the beginning and end of each string and the argument location within each sting at step 170. At step 180, the executable code stores the string constants and argument numbers and retains the location of the string constants and argument numbers in memory, for example, by assigning pointers to data structures containing the string ID or argument number. The data structures include a string ID and pointer or an argument number and pointer. At step 190, the executable code determines if a request has been received from a client. If a request has been received from a client (YES), the executable component proceeds to step 195. At step 195, the executable component executes any code relating to the request, loads the strings (*e.g.*, into an HTML page), inserts the arguments into the strings and transmits the requested information back to the requestor. The executable component then returns to step 190. If a request has not been received from a client (NO) at step 190, the executable component determines if any changes have occurred in any of the string template pages at step 200. If a change has occurred in any of the string template pages (YES), the executable component reloads and parses the modified

string template pages and retains the new string IDs and assigns new pointers to the data structure. The executable component then returns to step 190. If a change has not occurred in any of the string template pages (NO), [[T]] the executable component returns to step 190.